Personalizing Online Computer Engineering Resources and Labs for Digital, Embedded, and Computer System Courses

Peter Jamieson
Department of ECE
Miami University
Email: jamiespa@miamioh.edu

Ricardo Ferreira, José Augusto Nacif
Universidade Federal de Viçosa, Brazil.
Email: ricardo, jnacif @ufv.br

***Abstract*—The immediate and new challenges of the current Covid-19 pandemic have made it hard for all of us; in this work in progress as an innovative practice, we look to both leverage the challenges and share our work so that others might see some benefit to these times and improve their courses. In particular, our focus is on creating automated tools to physically create exams and sample code for Digital Systems and Computer Architecture courses. Additionally, we focus on shifting, traditional in-person labs to online, personalized formats for Digital Systems and Embedded Systems so that both educator and learner can still provide/experience virtual computer engineering education.**

We focus on three courses (Digital System Design, Computer Architecture/Organization, and Embedded System Design) as they are fundamentally driven by the implementation and execution of "algorithms". From this starting point, we have created tools to generate sample code and exams, and have found means to virtualize labs and hands-on activities. In particular, we have created Python tools that allow educators to personalize code and problems, create these codes/problems (as text files or incorporated in word documents), and email these documents to students. This provides the means to create problems and code examples that are different from their peers and can be assessed on a per individual basis to alleviate some of the challenges with live and proctored exams. Additionally, we have found tools and methods for students to virtually perform the hands-on portion of these three subjects without the need for traditional lab equipment. This requires students to spend less than 100 USD worth of equipment and software.

Our goal is to share these resources and our methodologies to help in this time of crisis. Additionally, these tools and methods have forced us to innovate our teaching, and we will, likely, use these tools and methods in the future. We share these tools in hope that the computer engineering education community will join this process to help us all improve our student's education.

I. Introduction

Remote labs have been an area of study that various researchers in a broad range of situations have studied alongside communication advancements. Most of us educators, however, were not actively engaged in this research until March 2020 when many of our schools went to remote learning and we were asked to rapidly adapt to a new norm. At our universities, our first efforts looked to either leverage simulation software, to send equipment off to students in their respective locations, or to eliminate aspects of courses that did not fit the new model. In the Americas, most of us had a few months to prepare for the following semester and the high likelihood of more remote teaching. In this work, we describe the tools we leveraged to maintain our hands-on components of our courses in the hopes of starting and joining in a conversation of what tools are available for computer engineering educators, and what innovation may our current crisis lead us to develop.

In addition to hands-on labs needing to be remotely done, we also needed to find a way to create individualized exam questions and code examples/problems for our students so that we could maintain our project-based (problem-based) engineering education approach for our remote students. Fortunately, computing educators have the expertise to create such tools, and in the fall and spring of 2020, we developed tools to create code samples, embed these in document files, and deliver these files to students via their university email accounts. This allows us to individualize assessment meaning we do not have to rely on cheating detection software (as much) or virtual proctoring tools. This approach will help maintain our hands-on learning, which engineering is well suited for, and we can stay away from the emerging lawsuit and privacy challenges [1].

We are documenting our process and presenting our approaches for two reasons. First, we believe that computer engineering educators (and beyond) need to share resources in some form of public space and provide a context of how their tools are used including release of open-source tools. This paper will, hopefully, start a meaningful discussion that will motivate us to create a virtual space within either IEEE or ACM for educators and our discussions and technical support for educational tools. Second, the crisis has forced us to innovate how we deliver our courses, and this push has resulted in practices that we believe are better than what we did before. The question remains, how can we evaluate our new approaches to verify if they are serving the students better in their learning? We argue that the question of "Is online/remote learning better/worse than in-person learning?" is a fundamental question that all of us now have a better understanding of, but we do not address this question in this paper.

The remainder of this paper is organized as follow: Section II introduces research in remote labs and individualized assess-

ment artifacts. Section III describes each of the three course areas, the method we employed for student's hands-on work, and the tools created to provide individualized assessment and learning artifacts. Finally, section IV provides additional discussion and concludes this paper.

II. Background - Remote Labs and Individualized Automated Problems

Given the Covid-19 pandemic, many higher education institutions have been exploring remote learning as an option to keep staff and students safer. Research papers are already emerging in this Covid-19 education space, for example, Shoufan's paper on teaching remote embedded systems [2]. From this emerging situation, a large group of us who teach courses with a hands-on element, typically in the form of labs, have had to innovate on how these portions of our courses are taught. This in turn has led to many articles, both positive [3] and negative [4], where researchers are wondering how this "new" method of learning is impacting the learners. However, both online education and remote labs are not a "new" research topic, and even though many of us are forced into this space for the first time, this has been an educational research space for some time now. In this section, we will, briefly, look at research in remote labs and investigate what techniques have been researched for automatically generating individual assessment artifacts. Also note, online teaching has been a research topic for many years, but we do not include a background in this space as it is not directly related to the topics in this work (we recommend John *et. al.* [5] as a starting point to view the research for online engineering education).

A. Remote and Simulation Labs

Hands-on experiences are a formal part of engineering education and one classification approach uses three areas as related to labs [6]:

1) Development Lab - as associated with project-based learning, a student designs to solve and to evaluate a system.
2) Research Lab - an open-ended exploration where learners use equipment to discover or understand a system.
3) Educational lab - learners come in with an understanding of a theory and experimentally verify this theory.

For undergraduate engineering, both development and educational labs are most commonly experienced where research is, mostly, a graduate degree experience.

For hands-on experiences, we define remote labs as:

- Shared-remote - learners share a lab resource via the internet when the lab resource is expensive and would, likely, be shared in an equivalent live lab [7].
- Station-remote - learners use a lab device remotely where each lab group gets their real lab equipment that is accessed remotely [7].
- Equipment-local - learners are either given or buy the lab equipment and perform the lab in their space [7].
- Simulation-local - learners run their labs via simulation software in a virtual setting [6].

This terminology is taken from 2 resources, and Deniz *et. al.*[7] describe features for these labs examining real, remote, and virtual labs based on "Feeling of reality", "Technical Support", etc. Striegel's remote lab work in 2001 appears to be one of the earliest efforts in remote labs for computer engineering courses [8], and they focused on a case in Iowa State University and evaluating the remote lab in terms of cost, licensing, labor, and learning.

We propose to provide our tools and setups for the following three courses citing some examples of remote and virtual lab research provided in each domain:

- Digital System Design - [9], [10]
- Computer Architecture/Organization - [11], [12] and related simulators [13]
- Embedded System Design - [14], [15]

Note that, this is not a full look at remote and virtual labs in any of these fields.

B. Automated Examples and Testing

The idea of automatic and customized assessment, examples, and tutoring has been a research goal that has been developed directly in tandem with computing technologies. For example, Hollingsworth [16] in 1960 published an article discussing the automated grading of programming assignments on punch cards. We classify automation of assessment into tutors, formal assessment artifacts and feedback, and examples.

The hope for automated tutors is to provide a system that can help the learner based on their current level of competency and provide the learner with examples, activities, and feedback to help them progress. Formally, these systems are called Intelligent Tutor Systems [17], and they have been designed for many decades. Many of these systems focus on computer science [18] as the teachers are themselves capable of creating the systems. However, as these technologies develop they have been adopted and applied to a range of academic fields [19], [20].

The second key desire for automation within education is a means to create and provide feedback on assessment artifacts. For example, Hollingsworth's system [16] is an example of an automated testing system for student's programs. Most of us are familiar with Scantron based grading for multiple-choice tests. There is a multitude of research in this space, and Douce *et. al.* 2005 review [21] looks at automatic testing for programming. The recent popularity of MOOCs has pushed some researchers to develop better customized assessment systems [22], and these systems have used many developmental languages such as Python [23] and Matlab [24]. We note that this work focuses on Python.

Finally, the last type of automated pedagogical tool that we focus on in this work is automated examples for learners. In particular, we look at how to create custom code snippets similar to Matsumoto *et. al.* work on generating C source code [25].

TABLE I
SOFTWARE AND HARDWARE SETUP FOR HAND-ON EXPERIENCES FOR REMOTE LEARNING

Class	Software	Hardware	Simulation	Price	Use	Technical Support
DSD-Miami	Quartus 16.1	DE2-115 (live lab)	ModelSim (remote lab)	Free	Local/Remote	Medium
DSD-UFV	Google Colab	-	Icarus Verilog (remote lab)	Free	Local/Remote	Low
DSD-UFV	DigitalJS	-	Yosys (remote lab)	Free	Local/Remote	Low
CA-Miami	MARs	-	-	Free	Local	Low
CA-UFV	WebRISC-V	-	-	Free	Local	Low
CA-Miami	RARs	-	-	Free	Local	Low
CA-Miami	VirtualBox	-	-	Free	Local	Low
	Ubuntu	-	-	Free	Local	Low
CA-Miami	Quartus 16.1	-	ModelSim	Free	Local/Remote	Low
CA-UFV	Google Colab	-	Icarus Verilog (remote lab)	Free	Local/Remote	Low
CA-Miami	AVR toolchain	ATMega328P	-	20 USD	Local	Low
ESD-Miami	AVR toolchain	ATMega328P	-	100 USD	Local	Medium
ESD-Miami	VirtualBox	-	-	Free	Local	Low
	Ubuntu	-	-	Free	Local	Low
ESD-UFV	-	-	TinkerCad	Free	Local/Remote	Low

III. INFRASTRUCTURE FOR COURSES

Table I summarizes the software, hardware, and simulation software used for each of the three course areas. In column 1, we specify the area of teaching (DSD = Digital System Design, CA = Computer Architecture, and ESD = Embedded System Design) and attach either Miami University (Miami) or Universidade Federal de Viçosa (UFV) to indicate at which institution the tools are used at. Columns 2 to 4 show the software name, hardware, and simulation software where a "-" indicates that there is nothing for this category. Column 5 shows an approximate cost in US dollars where "Free" means that the tools are either open-source or are provided free-of-use by a vendor. Column 6 describes how the tools are used where "Local" means that the students use the device where they are located and "Remote" means that a student logs onto a lab-hosted system. Finally, column 7 describes how much technical support a teacher needs to provide to the students.

Table II shows how we use Python scripts to generate artifacts for class (https://github.com/drpaj12/CpE_remote/wiki). Column 1 shows the class and uses the same notation as Table I, and columns 2 to 4 show if scripts were designed for creating exams, quizzes or assignments, and examples, respectively.

In the following sections, we will provide details for each class.

A. Digital System Design

Digital System Design focuses on introducing students to the transistor and how it can be used to create components for combinational and sequential circuits. Students learn how to use these components to create circuits that can be used to perform computation to a point where they can convert a simple sequential C program into a circuit implementation in a Hardware Description Language (HDL). Because FPGAs are, relatively, cheap to purchase and use, students can implement these circuits on FPGA prototyping boards like the DE2-115 from Terasic Inc. that we, normally, use in our live lab.

For our classes this year, Miami and UFV labs were only accessible by a small percentage of the class, and instead, we shifted our labs to simulation using ModelSim that comes with the free download of Intel's Quartus CAD software. This allows students to build and test their circuits, and the learning was focused more on creating testbenches to test their circuit designs. The only experience in the labs that were missed out on was mapping their circuit designs to FPGA pins that are then connected to peripherals such as switches, LEDs, and other integrated chips. The only part of the Miami course that was not included was the open-ended projects that we usually include [26] since the students did not have access to the FPGA prototyping boards.

The DSD-UFV course uses a set of Google Colab extensions [27], including a set of design examples and assignments, to teach logic circuit design, Verilog language, processor, and even use GPU architectures. These cloud labs also include the Valgrind and Gem5 computer architecture simulators. The browser-based DigitalJS [28] is used to introduce Verilog simple examples as well MIPS/RISC-V design [29].

Miami's course uses exams to assess students on their understanding of the material, and we created scripts to auto-generate exam questions such that each student had a unique instance of a question. This allowed us to have take-home exams that were similar, but unique enough that we did not worry about copying.

B. Computer Architecture/Organization

Computer architecture and organization courses have hands-on assignments and tests that can include students learning to create assembly language programs, build architectures, and implement optimizations to understand the low-level design and use of a simple processor.

At Miami, our architecture course is badge-based [30] meaning that after students complete a set of base badges that focus on assembly, architecture, and optimization they can pursue their interests in hardware implementation, assem-

TABLE II
PYTHON TOOLS CREATED TO GENERATE EXAMS, QUIZZES, AND EXAMPLES

Class	Exams	Quizzes/Assignments	Examples
DSD-Miami	Yes - Verilog HDL, C-code, Questions	No	No
CA-Miami	No	Yes - Optimizable C-code	Yes - C-code
ESD-Miami	No	No	Yes - C-code

bly programming, compiler design, operating system design, and/or security as all relate to computer organization. For this reason, the tools used for this course are broad (as listed in Table I), but all students use MARs [13] to start learning MIPs assembly programming. The other tools listed in the table are used depending on how a student chooses to proceed. For example, in the past few years around 2 to 5 students per year have chosen to implement a RISC-V processor on an FPGA [31] and use both Quartus as described in the previous section and RARs https://github.com/TheThirdOne/rars to create and test the programs for their processor.

Colab extensions are used in the CA-UFV approach for architecture simulations (Gem5), and a focus of teaching assembly language is facilitated with the browser-based WebRisc-V [32] software so that students can implement their hands-on programs.

For Miami's architecture course, we created Python scripts to create example C programs that students can take a C program and can translate the C into MIPs assembly code to understand what the compiler is doing, how control flow is implemented at a low-level, and how to use memory and addresses. We also create two C programs that students need to translate into assembly, and then they use their assembly implementation to understand how an architecture with a small cache or 4 stage pipeline would speedup the program execution compared to a base architecture.

C. Embedded Systems

Embedded system design has been quoted as: "Embedded Systems isn't well-grounded in fundamental concepts; rather, it often serves as the application of all the above concepts EE classes you've taken and will take into real-world systems. And it's not because embedded systems are the end-all/be-all of electrical engineering — rather, because embedded systems are the simplest real-world examples of these fundamental principles of EE." Jay Carlson [33].

From our perspective, there are two approaches to teaching embedded systems - 1) an open-ended project-based system design where students learn about their microcontrollers and peripherals to create their system 2) a low-level bottom-up understanding of a specific microcontroller and how it interacts with peripherals. Both approaches have benefits and costs, and we have discussed the merits of the first approach that mirrors that of a Maker space approach [34] [35]. However, this approach does not fit well within the remote setting, and for this space, we adopted a low-level bottom-up approach using the ATmega328P as our example microcontroller. The advantage of this device is that it is the processor on the Arduino UNO board, but we can bypass the Arduino IDE tools and use an open-source AVR-toolchain to program the device in C. We chose this processor since the prototyping board is cheap (20USD) and most students already have the board.

ESD-UFV uses Tinkercad (https://www.tinkercad.com/) to introduce basic laboratories by using virtual components, breadboard, wave generator, and oscilloscope. This allows students to build and simulate systems using virtual tools that are approximations of their real tools. This includes the possibility of using an Arduino UNO.

At Miami, we used Python scripts to generate simple C examples that demonstrate C programming concepts, but the majority of this course requires students to read and understand embedded system C code, and our use of automated generated artifacts is minimal for this course.

IV. CONCLUSION

In this paper, we described the tools and setups we used in Digital System Design, Computer Architecture/Organization, and Embedded Systems Design to teach the hands-on components of these courses online. We provide the context of this work to the broad research base related to remote labs and automated artifact creation. We describe our tools and provide links to the ones we created to help others in this space, and to promote a larger community effort into sharing our approaches to help improve learning.

Additionally, much of the forced innovation to teaching that we have experienced has helped us improve our teaching approaches, and many of the ideas and approaches that we tested will now be included in our live teaching of these courses. For example, the deeper dive into testbench creation for digital circuits will be included as part of the regular class as this mimics what real designers perform in industry, and a learner should be familiar with these tools.

REFERENCES

[1] J. Mullin, "Student surveillance vendor proctorio files slapp lawsuit to silence a critic," *Electronic Frontier Foundation*, 2021. [Online]. Available: https://www.eff.org/deeplinks/2021/02/student-surveillance-vendor-proctorio-files-slapp-lawsuit-silence-critic

[2] A. Shoufan, "Active distance learning of embedded systems," *IEEE Access*, vol. 9, pp. 41 104–41 122, 2021.

[3] L. McKenzie, "Cautious optimism about teaching stem online," *Inside Higher Ed*, 2021. [Online]. Available: https://www.insidehighered.com/news/2021/03/11/faculty-still-harbor-concerns-about-teaching-stem-courses-online

[4] I. Chettri, "Stem students struggle to grasp content from online labs," *The GW Hatchet*, 2020. [Online]. Available: https://www.gwhatchet.com/2020/11/02/stem-students-struggle-to-grasp-content-from-online-labs/

[5] J. Bourne, D. Harris, and F. Mayadas, "Online engineering education: Learning anywhere, anytime," *Journal of Engineering Education*, vol. 94, no. 1, pp. 131–146, 2005.

[6] B. Balamuralithara and P. C. Woods, "Virtual laboratories in engineering education: The simulation lab and remote lab," *Computer Applications in Engineering Education*, vol. 17, no. 1, pp. 108–118, 2009.

[7] D. Z. Deniz, A. Bulancak, and G. Ozcan, "A novel approach to remote laboratories," in *33rd Annual Frontiers in Education, 2003. FIE 2003.*, vol. 1. IEEE, 2003, pp. T3E–T3E.

[8] A. Striegel, "Distance education and its impact on computer engineering laboratories," in *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193)*, vol. 2, 2001, pp. F2D–4.

[9] H.-D. Wuttke, K. Henke, and N. Ludwig, "Remote labs versus virtual labs for teaching digital system design," in *Proceedings of the Int. Conf. On Computer Systems and Technologies CompSysTech*, vol. 5, 2005, pp. 2–6.

[10] F. Morgan, S. Cawley, and D. Newell, "Remote fpga lab for enhancing learning of digital systems," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 5, no. 3, pp. 1–13, 2012.

[11] B. Fechner, J. Keller, and W. Schiffmann, *Evaluation of a virtual computer architecture lab*. FernUniversität in Hagen, 2005.

[12] M. Rico, J. Ramírez, D. Riofrío, M. Berrocal-Lobo, and A. de Antonio, "An architecture for virtual labs in engineering education," in *Proceedings of the 2012 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2012, pp. 1–5.

[13] K. Vollmar and P. Sanderson, "Mars: an education-oriented mips assembly language simulator," in *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, 2006, pp. 239–243.

[14] I. Angulo, L. Rodrìguez-Gil, and J. Garcìa-Zubìa, "Scaling up the lab: An adaptable and scalable architecture for embedded systems remote labs," *IEEE Access*, vol. 6, pp. 16 887–16 900, 2018.

[15] P. Anzhelika, G. Olga, E. Ivanov, A. Sokolyanskii, and S. Kurson, "Development and application of remote laboratory for embedded systems design," in *Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. IEEE, 2015, pp. 69–73.

[16] J. Hollingsworth, "Automatic graders for programming classes," *Communications of the ACM*, vol. 3, no. 10, pp. 528–529, 1960.

[17] D. Sleeman and J. S. Brown, *Intelligent tutoring systems*. London: Academic Press, 1982.

[18] J. R. Anderson and E. Skwarecki, "The automated tutoring of introductory computer programming," *Communications of the ACM*, vol. 29, no. 9, pp. 842–849, 1986.

[19] B. P. Woolf, *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.

[20] E. O'Rourke, E. Butler, A. D. Tolentino, and Z. Popović, "Automatic generation of problems and explanations for an intelligent algebra tutor," in *International Conference on Artificial Intelligence in Education*. Springer, 2019, pp. 383–395.

[21] C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," *Journal on Educational Resources in Computing (JERIC)*, vol. 5, no. 3, pp. 4–es, 2005.

[22] S. Miranda, G. R. Mangione, F. Orciuoli, M. Gaeta, and V. Loia, "Automatic generation of assessment objects and remedial works for moocs," in *2013 12th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 2013, pp. 1–8.

[23] C. Andujar, "Procedural generation of stem quizzes," *arXiv preprint arXiv:2009.03868*, 2020.

[24] M. Fikar, M. Bakošová, T. Hirmajer *et al.*, "Automatic generation of assignments and quizzes in control engineering education," in *2007 European Control Conference (ECC)*. IEEE, 2007, pp. 2714–2720.

[25] S. Matsumoto, K. Okimoto, T. Kashima, and S. Yamagishi, "Automatic generation of c source code for novice programming education," in *International Conference on Human-Computer Interaction*. Springer, 2016, pp. 65–76.

[26] P. Jamieson, "Early project based learning improvements via a star trek engineering room game framework, and competition," in *Frontiers in Education Conference (FIE), 2011*, oct. 2011, pp. S4H–1 –S4H–5. [Online]. Available: http://www.users.muohio.edu/jamiespa/html_papers/fie_11.pdf

[27] M. Canesche, L. Silva, O. P. V. Neto, J. A. Nacif, and R. Ferreira, "Google colab cad4u: Hands-on cloud laboratories for digital design," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[28] M. Materzok, "Digitaljs: A visual verilog simulator for teaching," in *Proceedings of the 8th Computer Science Education Research Conference*, 2019, pp. 110–115.

[29] F. Passe, M. Canesche, O. P. V. Neto, J. A. Nacif, and R. Ferreira, "Mind the gap: Bridging verilog and computer architecture," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[30] P. Jamieson, "Does badge-based learning buck the grading curve? an educational experiment in computer architecture," in *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2014, p. 1.

[31] T. McGrew and E. Schonauer, "Framework and tools for undergraduates designing risc-v processors on an fpga in computer architecture education," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2019, pp. 778–781.

[32] R. Giorgi and G. Mariotti, "Webrisc-v: A web-based education-oriented risc-v pipeline simulation environment," in *Proceedings of the Workshop on Computer Architecture Education*, 2019, pp. 1–6.

[33] J. Carlson, "How i teach embedded systems," *jaycarlson*, 2019. [Online]. Available: https://jaycarlson.net/2019/07/26/how-i-teach-embedded-systems/

[34] P. Jamieson and J. Herdtner, "More missing the boat - arduino, raspberry pi, and small prototyping boards and engineering education needs them," in *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE*. IEEE, 2015, pp. 1–6.

[35] P. Jamieson, "Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat?" in *Proc. FECS*, 2010, pp. 289–294.